

# The Treasure beneath Convolutional Layers: Cross-convolutional-layer Pooling for Image Classification\*

Lingqiao Liu<sup>1</sup>, Chunhua Shen<sup>1,2</sup>, Anton van den Hengel<sup>1,2</sup>

<sup>1</sup> The University of Adelaide, Australia

<sup>2</sup> Australian Centre for Robotic Vision

## Abstract

A number of recent studies have shown that a Deep Convolutional Neural Network (DCNN) pretrained on a large dataset can be adopted as a universal image description which leads to astounding performance in many visual classification tasks. Most of these studies, if not all, adopt activations of the fully-connected layer of a DCNN as the image or region representation and it is believed that convolutional layer activations are less discriminative.

This paper, however, advocates that if used appropriately convolutional layer activations can be turned into a powerful image representation which enjoys many advantages over fully-connected layer activations. This is achieved by adopting a new technique proposed in this paper called cross-convolutional-layer pooling. More specifically, it extracts subarrays of feature maps of one convolutional layer as local features and pools the extracted features with the guidance of feature maps of the successive convolutional layer. Compared with existing methods that apply DCNNs in the local feature setting, the proposed method is significantly faster since it requires much fewer times of DCNN forward computation. Moreover, it avoids the domain mismatch issue which is usually encountered when applying fully connected layer activations to describe local regions. By applying our method to four popular visual classification tasks, it is demonstrated that the proposed method can achieve comparable or in some cases significantly better performance than existing fully-connected layer based image representations while incurring much lower computational cost.

## Contents

### 1. Introduction 2

2. Current strategies for creating image representations from a pretrained DCNN	2
3. Proposed method	3
3.1. Convolutional layer vs. fully-connected layer	3
3.2. Cross-convolutional-layer pooling . . . . .	4
3.3. Creating finer resolutions of spatial units partitioning . . . . .	5
4. Experiments	6
4.1. Experimental protocol . . . . .	6
4.2. Performance evaluation . . . . .	6
4.2.1 Classification results . . . . .	6
4.2.2 Computational cost . . . . .	8
4.3. Analysis of components of our method . . .	8
4.3.1 Using different convolutional layers	9
4.3.2 Comparison of different pooling schemes . . . . .	9
4.3.3 Feature sign quantization . . . . .	9
5. Conclusion	9

\*This work is in part supported by ARC Grants FT120100969, and LP130100156. Correspondence should be addressed to C. Shen (email: chhshen@gmail.com).

## 1. Introduction

Recently, Deep Convolutional Neural Networks (DCNNs) have attracted a lot of attention in visual recognition due to its good performance [15]. It has been discovered that activations of a DCNN pretrained on a large dataset, such as ImageNet [6], can be employed as a universal image representation and applying this representation to many visual classification problems leads to astounding performance [21, 2]. This discovery quickly sparks a lot of interest and inspires a number of further extensions [12, 16]. A fundamental issue of this kind of methods is that how to generate image representation from a pretrained DCNN. Most of current solutions, if not all, take activations of the fully connected layer as the image representation. In contrast, activations of convolutional layers are rarely used and some studies [23, 1] have reported that directly using convolutional layer activations as image features produces inferior performance.

In this paper, however, we advocate that convolutional layer activations can be turned into a powerful image representation if they are used appropriately. We propose a new method called cross-convolutional layer pooling, or cross layer pooling in short, to fully leverage the discriminative information of convolutional layers. The main contributions and also key differences to the previous attempt of using convolutional layer activations lay in two aspects: (1) we use convolutional layer activations in a ‘local feature’ setting which extracts subarrays of convolutional layer activations as region descriptors. (2) we pool extracted local features with cross-layer information.

The first aspect is motivated by some recent works [12, 16] which have shown that DCNN activations are not translational invariant and it is beneficial to apply a DCNN to describe local regions and create the image representation by pooling multiple regional DCNN activations. Our method steps further to use subarrays of convolutional layer activations, that is, parts of CNN activations as regional descriptors. Compared with previous works [12, 16], our method enjoys two major advantages: (1) instead of running DCNN forward computation multiple times, one for each local region, our method only needs to run a DCNN once (or very few times in our multiple-resolution scheme) for all local regions. This results in great computational cost saving. (2) existing methods [12, 16] essentially apply a network trained for representing an image to represent a local region. This causes significant domain mismatch between the input at the training stage and testing stages, which may degrade the discriminative power of DCNN activations. In contrast, our method avoids this issue since it still uses the whole image as the network input and only extracts parts of convolutional activations as regional descriptors.

The second aspect is motivated by the parts-based pooling methods [17, 26, 24] which are commonly used in fine-

grained image classification. This kind of methods create one pooling result for each detected part region and the final image representation is obtained by concatenating pooling results from multiple parts. We generalize this idea into the context of DCNN and avoid using any predefined parts annotation. More specifically, we deem the feature map of each filter in a convolutional layer as the detection response map of a part detector and apply the feature map to weight regional descriptors extracted from previous convolutional layer in the pooling process. The final image representation is obtained by concatenating pooling results from multiple channels with each channel corresponding to one feature map. Note that unlike existing regional-DCNN based methods [12, 16], the proposed method does not need any additional dictionary learning and encoding steps at both training and testing stages. Besides the above two aspects, we also develop a simple scheme to extract local features at a finer resolution from convolutional layers and experiment with a coarse feature quantization scheme which significantly reduces the memory usage in storing image representations.

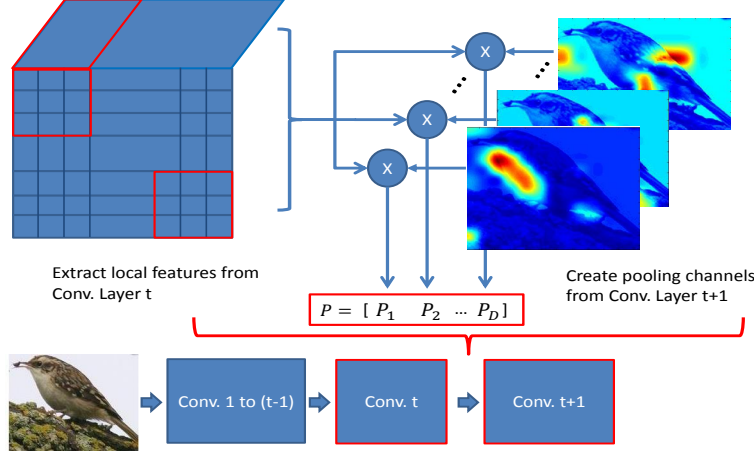
We conduct extensive experiments on four datasets covering four popular visual classification tasks, that is, scene classification, fine-grained object classification, generic object classification and attribute classification. Experimental results suggest that the proposed method can achieve comparable or in some cases significantly better performance than competitive methods while being considerably faster in creating image representations.

**Preliminary:** Our network structure and model parameters are identical to those in [15], that is, we have five convolutional layers and two fully connected layers. We use conv-1, conv-2, conv-3, conv-4, conv-5, fc-6, fc-7 to denote them respectively. At each convolutional layer, multiple filters are applied and it results in multiple feature maps, one for each filter. In this paper, we use the term ‘feature map’ to indicate the convolutional results (after applying the ReLU) of one filter and the term ‘convolutional layer activations’ to indicate feature maps of all filters in a convolutional layer.

## 2. Current strategies for creating image representations from a pretrained DCNN

In the literature, there are two major ways of using a pretrained DCNN to extract image representations: using a pretrained DCNN as global features and using a retrained DCNN as local features.

The first way takes the whole image as the input to a pretrained DCNN and extracts the fc-6/fc-7 activations as the image-level representation. To make the network better adapted to a given task, fine-tuning sometimes is applied. Also, to make this kind of methods more robust toward image transforms, averaging activations from several jittered versions of the original image, e.g. a slightly shifted version



**Figure 1:** An overview of the proposed method.

of the input image or a mirrored version of the input image, has been employed to obtain better classification performance [2].

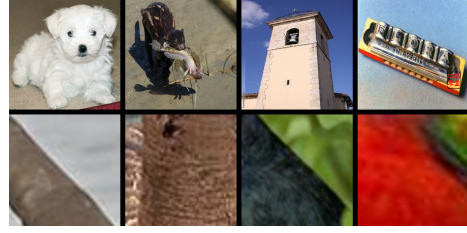
DCNNs can also be applied to extract local features. It has been suggested that DCNN activations are not invariant to a large amount of translation [12] and the performance can be degraded if input images are not well aligned. To handle this, it is suggested to sample multiple regions from an input image and use one DCNN, called regional-DCNN in this scenario, to describe each region. The final image representation is aggregated from activations of those regional-DCNNs. Usually, another layer of unsupervised encoding is employed to create the image-level representation [12, 16]. Also, multiple-scale extraction strategy can be applied to further boost performance [12]. It has been shown that for many visual tasks [12, 16] this kind of methods lead to better performance than directly extracting DCNN activations as global features.

One common factor of the above methods is that they all use fully-connected layer activations as features. The convolutional layer activations, however, are not usually employed and some preliminary studies [23, 1] have suggested that the convolutional layer has weaker discriminative power than the fully-connected layer.

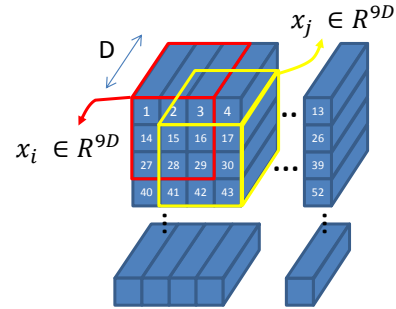
### 3. Proposed method

#### 3.1. Convolutional layer vs. fully-connected layer

One major difference between convolutional and fully-connected layer activations is that the former is embedded with rich spatial information while the latter does not. The convolutional layer activations can be formulated as a tensor of the size  $H \times W \times D$ , where  $H, W$  denote the height and width of each feature map and  $D$  denotes the number of feature maps. Essentially, the convolutional layer divides the input image into  $H \times W$  regions and uses  $D$ -dimensional



**Figure 2:** This figure demonstrates the domain mismatch issue when applying the fully-connected layer activations as regional descriptors. Top row: input images that a DCNN 'sees' at the training stage. Bottom row: input images that a DCNN 'sees' at the test stage.



**Figure 3:** The demonstration of extracting local features from a convolutional layer.

feature maps to describe the visual pattern within each region. Thus, convolutional layer activations can be viewed as a 2-D array of  $D$ -dimensional *local features* with each one describing a local region. For the clarity of presentation, we call each of the  $H \times W$  regions as a **spatial unit**, and the  $D$ -dimensional feature maps corresponding to a spatial unit as the **feature vector in a spatial unit**. The fully-

connected layer takes the convolutional layer activations as the network input and transforms them into a feature vector representing the whole image. In this process, the spatial information is lost and the feature vector in a spatial unit cannot be explicitly recovered from activations of a fully-connected layer.

As mentioned in section 2, DCNNs can also be applied to extract local features to handle the drawback of being translational variant. However, this scheme comes along with two side-effects in practice: (1) Its computational cost becomes higher than using DCNN activations as global image features since in this scheme one needs to run DCNN forward computing multiple times, one for each region. (2) Moreover, it makes the input of a DCNN become significantly different from the input images that are used to train the network. This is because when applied as a regional feature, a DCNN is essentially used to describe local visual patterns which correspond to small parts of objects rather than whole images at the training stage. In Figure 2, we plot some training images from the ImageNet dataset and resized local regions. As can be seen, although they all have the same image size, their appearance and levels of details are quite different. Thus, blindly applying fully-connected layer activations as local features introduces significant domain mismatch which could potentially undermine the discriminative power of DCNN activations.

Our idea to handle aforementioned drawbacks is to extract multiple regional descriptors from *a single DCNN applied to a whole image*. We realize this idea by leveraging the spatial information of convolutional layers. More specifically, in convolutional layers, we can easily locate a subset of activations which correspond to a local region. These subset of activations correspond to a set of subarrays of convolutional layer activations and we use them as local features. Figure 3 demonstrates the extraction of such local features. For example, we can first extract  $D$ -dimensional feature vectors from regions 1, 2, 3, 14, 15, 16, 27, 28, 29 and concatenate them into a  $9 \times D$ -dimensional feature vector and then shift one unit along the horizontal direction to extract features from regions 2, 3, 4, 15, 16, 17, 28, 29, 30. After scanning all the  $13 \times 13$  feature maps we obtain 121 (omitting boundary spatial units)  $(9 \times D)$ -dimensional local features.

It is clear that the proposed method enjoys the following merits: (1) the input of the DCNN is still a whole image rather than local regions. Thus the domain mismatch issue is avoided. (2) we only need to run DCNN forward calculation once and thus computational cost can be greatly saved. Note that in our method, we extract regional features from multiple spatial units and concatenate the feature vectors from them. This is different to previous work [27] (although their method is for a different application) which only treats the feature vector in one spatial unit as the local

feature. We find that the use of feature vectors from multiple spatial units can significantly boost classification performance. This is because the feature vector from a single spatial unit may not be descriptive enough to characterize the visual pattern within a local region.

### 3.2. Cross-convolutional-layer pooling

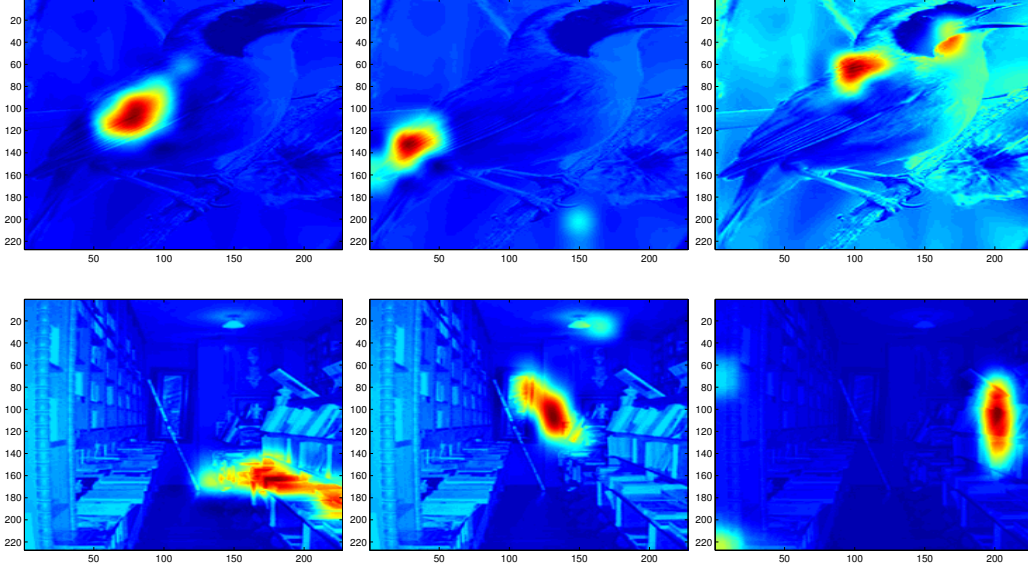
After extracting local features from a convolutional layer, one can directly perform traditional max-pooling or sum-pooling to obtain the image-level representation. In this section, we propose an alternative pooling method which can significantly improve the classification performance. The proposed method is inspired by the parts based pooling strategy [17, 26, 24] in fine-grained image classification. In this kind of methods, multiple regions-of-interest (ROI) are firstly detected and each of them corresponds to one human-specified object part, e.g. the tail of birds. Then local features falling into each ROI are pooled together to obtain a pooled feature vector. Given  $D$  object parts, this strategy creates  $D$  different pooled feature vectors and these vectors are concatenated together to form the final image representation. It has been shown that this simple strategy achieves significantly better performance than blindly pooling all local features together. Formally, the pooled feature from the  $k$ th ROI, denoted as  $\mathbf{P}_k^t$ , can be calculated by the following equation (let's consider sum-pooling in this case):

$$\mathbf{P}_k^t = \sum_{i=1} \mathbf{x}_i I_{i,k}, \quad (1)$$

where  $\mathbf{x}_i$  denotes the  $i$ th local feature and  $I_{i,k}$  is a binary *indicator map* indicating that if  $\mathbf{x}_i$  falls into the  $k$ th ROI. We can also generalize  $I_{i,k}$  to real value with its value indicating the ‘membership’ of a local feature to a ROI. Essentially, each indicator map defines a pooling channel and the image representation is the concatenation of pooling results from multiple channels.

However, in a general image classification task, there is no human-specified parts annotation and even for many fine-grained image classification tasks, the annotation and detection of these parts are usually non-trivial. To handle this situation, in this paper we propose to use feature maps of the  $(t+1)$ th convolutional layer as  $D_{t+1}$  indicator maps. By doing so,  $D_{t+1}$  pooling channels are created for the local features extracted from the  $t$ th convolutional layer. We call this method cross-convolutional-layer pooling or cross-layer pooling in short. The use of feature maps as indicator maps is motivated by the observation that a feature map of a deep convolutional layer is usually sparse and indicates some semantically meaningful regions<sup>1</sup>. This observation is illustrated in Figure 4. In Figure 4, we choose two images taken from two datasets, Birds-200 [10] and MIT-67

<sup>1</sup>Note that similar observation has also been made in [23].



**Figure 4:** Visualization of some feature maps extracted from the 5th layer of a DCNN.

[20]. We randomly sample some feature maps from 256 feature maps in conv5 and overlay them to original images for better visualization. As can be seen from Figure 4, the activated regions of the sampled feature map (highlighted in warm color) are actually semantically meaningful. For example, the activated region in top-left corner of Figure 4 corresponds to the wing-part of a bird. Thus, the filter of a convolutional layer works as a part detector and its feature map serves a similar role as the part region indicator map. Certainly, compared with the parts detector learned from human specified part annotations, the filter of a convolutional layer is usually not directly task-relevant. However, the discriminative power of our image representation can be benefited from combining a much larger number of indicator maps, e.g. 256 as opposed to 20-30 (the number of parts usually defined by human), which is akin to applying bagging to boost the performance of multiple weak classifiers.

Formally, image representation extracted from cross-layer pooling can be expressed as follows:

$$\mathbf{P}^t = [\mathbf{P}_1^t, \mathbf{P}_2^t, \dots, \mathbf{P}_k^t, \dots, \mathbf{P}_{D_{t+1}}^t]$$

$$\text{where, } \mathbf{P}_k^t = \sum_{i=1}^{N_t} \mathbf{x}_i^t a_{i,k}^{t+1}, \quad (2)$$

where  $\mathbf{P}^t$  denotes the pooled feature for the  $t$ -th convolutional layer, which is calculated by concatenating the pooled feature of each pooling channel  $\mathbf{P}_k^t, k = 1, \dots, D_{t+1}$ .  $\mathbf{x}_i^t$  denotes the  $i$ th local feature in the  $t$ th convolutional layer. Note that feature maps of the  $(t+1)$ th convolutional layer is obtained by convolving feature maps of the  $t$ th convolutional layer with a  $m \times n$ -sized kernel. So if we extract

local features  $\mathbf{x}_i^t$  from each  $m \times n$  spatial units in the  $t$ th convolutional layer then each  $\mathbf{x}_i^t$  naturally corresponds to a spatial unit in the  $(t+1)$ th convolutional layer. Let's denote the feature vector in this spatial unit as  $\mathbf{a}_i^{t+1} \in \mathbb{R}^{D_{t+1}}$  and the value at its  $k$ th dimension as  $a_{i,k}^{t+1}$ . Then we use  $a_{i,k}^{t+1}$  to weight local feature  $\mathbf{x}_i^t$  in the  $k$ th pooling channel.

**Implementation Details:** In our implementation, we perform PCA on  $\mathbf{x}_i^t$  to reduce the dimensionality of  $\mathbf{P}^t$ . Also, we apply power normalization to  $\mathbf{P}^t$ , that is, we use  $\hat{\mathbf{P}}^t = \text{sign}(\mathbf{P}^t) \sqrt{|\mathbf{P}^t|}$  as the image representation to further improve performance. We also tried to directly use  $\text{sign}(\mathbf{P}^t)$  as image representations, that is, we coarsely quantize  $\mathbf{P}^t$  into  $\{-1, 1, 0\}$  according to the feature sign of  $\mathbf{P}^t$ . To our surprise, this only introduces a slight performance drop. This observation allows us to simply use 2-bits to represent each feature dimension and this saves a lot of memory to store image representations. Please refer to section 4.3.3 for more detailed discussion.

### 3.3. Creating finer resolutions of spatial units partitioning

One drawback of the above method is that it only works in a single resolution. For example, if the 4-th and 5-th convolutional layer are used, the image can only be partitioned into  $13 \times 13$  spatial units. For some applications such as scene classification, the object of interest is usually small and it is favorable to use finer partitioning to capture finer object details. To achieve this goal, we proposed to divide the whole image into multiple non-overlap or overlapped blocks and apply a DCNN to each block. Then same as in section 3.2, local features are extracted from the convo-

**Table 1:** Comparison of results on MIT-67. The lower part of this table lists some results reported in the literature. The proposed methods are denote with \*. For each method, the required number of CNN forward calculation is denoted as  $\text{CNN} \times k$ .

Methods	Accuracy	Comments
CNN-Global	57.9%	$\text{CNN} \times 1$
CNN-Jitter	61.1%	$\text{CNN} \times 10$
R-CNN SCFV [16]	68.2%	$\text{CNN} \times 100$
*CL-45	64.6%	$\text{CNN} \times 1$
*CL-45F	65.8%	$\text{CNN} \times 4$
*CL-45C	68.8%	$\text{CNN} \times 5$
*CL + CNN-Global	70.0%	$\text{CNN} \times 6$
*CL + CNN-Jitter	<b>71.5%</b>	$\text{CNN} \times 15$
Fine-tuning [2]	66.0%	fine-tuning on MIT-67
MOP-CNN [12]	68.9%	$\text{CNN} \times 53$ , three scales
VLAD level2 [12]	65.5%	$\text{CNN} \times 16$ , single scale
CNN-SVM [21]	58.4%	-
FV+DMS [7]	63.2%	-
DPM [18]	37.6%	-

lutional layer of each DCNN. In total, it generates much more spatial units for the whole image, for example, if we partition the whole image into four quadrats we will have  $26 \times 26$  spatial units in total. This scheme is illustrated as in Figure 5. Note that using this scheme we can easily create more spatial units by only introducing very few number of DCNN forward computation. In practice, we adopt a multi-resolution scheme which combines image representations extracted from multiple resolutions to achieve better classification performance.

## 4. Experiments

We evaluate the proposed method on four datasets: MIT indoor scene-67 (MIT-67 in short) [20], Caltech-UCSD Birds-200-2011 [10] (Birds-200 in short), Pascal VOC 2007 [11] (Pascal-07 in short) and H3D Human Attributes dataset [3] (H3D in short). These four datasets cover several popular topics in image classification, that is, scene classification, fine-grained object classification, generic object classification and attribute classification. Previous studies [21, 2] have shown that using activations from the fc layer of a pre-trained DCNN leads to surprisingly good performance in those datasets. Here in our experiments, we further compare different ways to extract features from a pretrained DCNN. We organized our experiments into two parts, the first part compares the proposed method with other competitive methods and the second part examines the impact of various components in our method.

### 4.1. Experimental protocol

We compare the proposed method against three baselines, they are: (1) directly using fully-connected layer activations for the whole image (CNN-Global); (2) averaging fully-connected layer activations from several transformed versions of the input image. Following [21, 2], we transform the input image by cropping its four corners and middle regions as well as by creating their mirrored versions; (3) the method in [21, 2] which extracts fully-connected layer CNN activations from multiple regions in an image and encodes them using sparse coding based Fisher vector encoding (RCNN-SCFV). Since RCNN-SCFV has demonstrated superior performance than the MOP method in [12], we do not include MOP in our comparison. To make fair comparison, we reimplement all three baseline methods. For the last method, we use the code provided by the author of [16] to extract the regional CNN features and perform encoding.

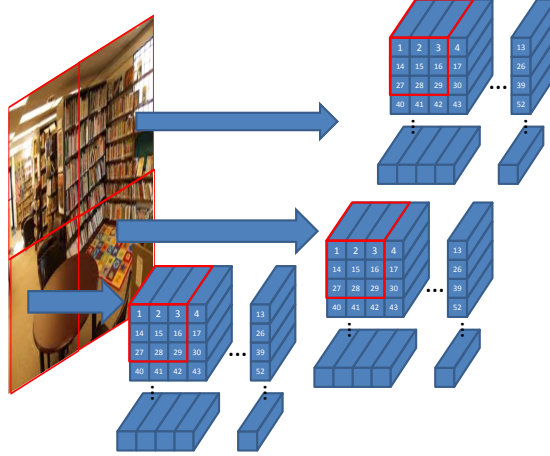
For our method, we use the multi-resolution scheme suggested in section 3.3, that is, besides applying our method to the whole image, we also partition the image into  $M \times N$  blocks and apply our method. The final image representation is the concatenation of image representations obtained from these two resolutions. For all datasets except H3D, we set  $M = N = 2$  and we set  $M = 2, N = 1$  for H3D because most images in H3D have longer height than width.

In the first part of our experiments, we report the result obtained by using the 4th and 5th convolutional layer since using them achieves the best performance. We denote our methods as CL-45, CL-45F, CL-45C, corresponding to the settings of applying our method to the whole image, to multiple blocks for finer resolution and combining representations from two different resolutions respectively. We also conduct similar experiment on the 3-4th layer of a DCNN in the second part of experiments and denote them as CL-34, CL-34F and CL-34C. To reduce the dimensionality of image representations, we perform PCA on local features extracted from convolutional layers and reduce their dimensionality to 500 before cross-layer pooling. In practice, we find that reducing to higher dimensionality only slightly increases the performance. We use libsvm [4] as the SVM solver and use precomputed linear kernels as inputs. This is because the calculation of linear kernels/Gram matrices can be easily implemented in parallel. When feature dimensionality is high this part of computation actually occupies most of computational time. Thus it is appropriate to use parallel computing to accelerate this part.

### 4.2. Performance evaluation

#### 4.2.1 Classification results

**Scene classification: MIT-67.** MIT-67 is a commonly used benchmark for evaluating scene classification algorithms, it



**Figure 5:** Our scheme to create finer spatial unit partition.

contains 6700 images with 67 indoor scene categories. Following the standard setting, we use 80 images in each category for training and 20 images for testing. The results are shown in Table 1. It can be seen that all the variations of our method (methods with ‘\*’ mark in Table 1) outperforms the methods that use DCNN activations as global features (CNN-Global and CNN-Jitter). This clearly demonstrates the advantage of using DCNN activations as local features. We can also see that the performance obtained by combining CL-45 and CL-45F, denoted as CL-C, has already achieved the same performance as the regional-CNN based methods (R-CNN SCFV and MOP-CNN) while it requires much fewer times of CNN forward calculation. Moreover, combining with the global-CNN representation, our method can obtain further performance gain. By combining CL-C with CNN-Jitter, our method, denoted as CL+CNN-Global and CL+CNN-Jitter respectively, achieves impressive classification performance 71.5%.

**Fine-grained image classification: Birds-200.** Birds-200 is the most popular dataset in fine-grained image classification research. It contains 11788 images with 200 different bird species. This dataset provides ground-truth annotations of bounding boxes and parts of birds, e.g. the head and the tail, on both the training set and the test set. In this experiment, we just use the bounding box annotation. The results are shown in Table 2. As can be seen, the proposed method performs especially well on this dataset. Merely CL-45 has already achieved 72.4% classification accuracy, winning 6% improvement over the performance of R-CNN SCFV which as far as we know is the best performance obtained in the literature when no parts information is utilized. Combining with CL-45F, our performance can be improved to 73.5%. It is quite close to the best performance obtained from the method that relies on strong parts annotation. Another interesting observation is that for this dataset, CL-45 signifi-

cantly outperforms CL-45F, which is in contrary to the case in MIT-67. This suggests that the suitable resolution of spatial units may vary from dataset to dataset.

**Object classification: Pascal-2007.** Pascal VOC 2007 has 9963 images with 20 object categories. The task is to predict the presence of each object in each image. Note that most object categories in Pascal-2007 are also included in ImageNet. So ImageNet can be seen as a super-set of Pascal-2007. The results on this dataset are shown in Table 3. From Table 3, we can see that the best performance of our method (CL + CNN-Jitter) achieves comparable performance to the state-of-the-art. Also, by merely using the feature extracted from convolutional layer, our method CL-45C outperforms the CNN-Global and CNN-Jitter which use DCNNs to extract global image features. However, our CL-45C does not outperform R-CNN and our best performed method CL + CNN-Jitter does not achieve significant performance improvement as what it has achieved in MIT-67 and Birds-200. This is probably due to that the 1000 categories in ImageNet training set has already included 20 categories in Pascal-2007. Thus the fully-connected layer actually contains some classifier-level information and using fully-connected layer activations implicitly utilizes more training data from ImageNet.

**Attribute classification: H3D.** In recent years, attributes of object, which are semantic or abstract qualities of object and can be shared by many categories, have gained increasing attention due to its potential use in zero/one-shot learning and image retrieval [19, 14]. In this experiment, we evaluate the proposed method on the task of predicting attribute of human. We use H3D dataset [3] which defines 9 attributes for a subset of ‘person’ images from Pascal VOC 2007 and H3D. The results are shown in Table 4. Again, our method shows quite promising results. Merely using the information from convolutional layer, our approach has al-



**Table 2:** Comparison of results on Birds-200. Note that the method with “use parts” mark requires parts annotations and detection while our methods do not employ these annotations so they are not directly comparable with us.

Methods	Accuracy	Remark
CNN-Global	59.2%	CNN $\times$ 1, no part.
CNN-Jitter	60.5%	CNN $\times$ 1, no part
R-CNN SCFV [16]	66.4%	CNN $\times$ 100, no part
*CL-45	72.4%	CNN $\times$ 1, no part
*CL-45F	68.4%	CNN $\times$ 4, no part
*CL-45C	<b>73.5%</b>	CNN $\times$ 5, no part
*CL + CNN-Global	72.4%	CNN $\times$ 6, no part
*CL + CNN-Jitter	73%	CNN $\times$ 16, no part
GlobalCNN-FT [2]	66.4 %	no parts, fine tuning
Parts-RCNN-FT [24]	76.37 %	use parts, fine tuning
Parts-RCNN [24]	68.7 %	use parts, no fine tuning
CNNaug-SVM [21]	61.8%	CNN $\times$ 1
CNN-SVM [21]	53.3%	CNN global
DPD+CNN [8]	65.0%	use parts
DPD [25]	51.0%	-

**Table 3:** Comparison of results on Pascal VOC 2007.

Methods	mAP	Remark
CNN-Global	71.7%	CNN $\times$ 1
CNN-Jitter	75.0%	CNN $\times$ 10
R-CNN SCFV [16]	76.9%	CNN $\times$ 100
*CL-45	72.6%	CNN $\times$ 1
*CL-45F	71.3%	CNN $\times$ 4
*CL-45C	75.0%	CNN $\times$ 5
*CL + CNN-Global	76.5%	CNN $\times$ 6
*CL + CNN-Jitter	<b>77.8%</b>	CNN $\times$ 15
CNNaug-SVM [21]	77.2%	with augmented data
CNN-SVM [21]	73.9%	no augmented data
NUS [22]	70.5%	-
GHM [5]	64.7%	-
AGS [9]	71.1%	-

**Table 4:** Comparison of results on Human attribute dataset.

Methods	mAP	Remark
CNN-Global	74.1%	CNN $\times$ 1
CNN-Jitter	74.6%	CNN $\times$ 10
R-CNN SCFV [16]	73.1%	CNN $\times$ 100
*CL-45	75.3%	CNN $\times$ 1
*CL-45F	70.7%	CNN $\times$ 4
*CL-45C	77.3%	CNN $\times$ 5
*CL + CNN-Global	78.1%	CNN $\times$ 6
*CL + CNN-Jitter	<b>78.3%</b>	CNN $\times$ 15
PANDA [26]	78.9	needs poselet annotation
CNN-FT [2]	73.8	CNN-Global, fine tuning
CNNaug-SVM [21]	73.0%	with augmented data
CNN-SVM [21]	70.8%	no augmented data
DPD [22]	69.9%	-

ready achieved 77.3% which outperforms R-CNN SCFV by 4%. By combining with CNN-Jitter, our method becomes comparable to PANDA [26] which needs complicated poselet annotations and detections.

#### 4.2.2 Computational cost

It is clear that our method requires much less time in DCNN forward computation. To give an intuitive idea of the computational cost incurred by our method, we report the average time spend on extracting image representations of various methods in Table 5. As can be seen, the computational cost of our method is comparable to that of CNN-

Global and CNN-Jitter. This is quite impressive given that our method achieves significantly better performance than these two methods. Compare with SCFV, the most competitive method to our approach in the sense of classification accuracy, we have obtained around 10 times speedup. Note that this speed evaluation is based on our naive MATLAB implementation and our method can be further accelerated by using C++ or GPU implementation.

#### 4.3. Analysis of components of our method

From the above experiments, the advantage of using the proposed method has been clearly demonstrated. In this section, we further examine the effect of various compo-



**Table 5:** Average time used for extracting an image representation for different methods. The time can be break down into two parts, time spend on extracting CNN features and time spend on performing pooling.

Method	CNN Extraction	Pooling	Total
*CL-45	0.45s	0.14s	0.6s
*CL-45F	1.3s	0.27s	1.6s
*CL-45C	1.75s	0.41s	2.2s
CNN-Global	0.4s	0s	0.4s
CNN-Jitter	1.8s	0s	1.8s
SCFV [16]	19s	0.3s	19.3s

**Table 6:** Comparison of results obtained by using different pooling layers.

Method	MIT-67	Birds200	Pascal07	H3D
CL-34	61.7%	64.6%	66.3%	74.7%
CL-34F	61.4%	61.4%	64.9%	70.4%
CL-34C	64.1%	66.8%	68.5%	75.9%
CL-45C	<b>68.8%</b>	<b>73.5%</b>	<b>75.0%</b>	<b>77.3%</b>

nents in our method.

#### 4.3.1 Using different convolutional layers

First, we are interested to examine the performance of using convolutional layers other than the 4th and 5th convolutional layers. We experiment with the 3th and 4th convolutional layers and report the resulting performance in Table 6. From the result we can see that using 4-5th layers can achieve superior performance than 3-4th layers. This is not surprising since it has been observed that the deeper the convolutional layer is, the better discriminative power it has [23].

#### 4.3.2 Comparison of different pooling schemes

The cross-layer pooling is an essential component in our method. In this experiment, we compare it against other possible alternative pooling approaches, they are: directly performing sum-pooling (with square operation) and max-pooling, using spatial pyramid pooling as suggested in [13], applying the sparse coding based fisher vector encoding (SCFV) [16] to encode extracted local features and perform pooling. To simplify the comparison, we only report results on the best performed single resolution setting for each dataset, that is, CL-45F for MIT-67 and CL-45 for the rest of three datasets. The results are shown in Table 7. As can be seen, the proposed cross-layer pooling significantly outperforms directly applying max-pooling or sum-pooling or even spatial-pyramid pooling. By applying an

**Table 7:** Comparison of results obtained by using different pooling schemes

Method	MIT-67	Birds200	Pascal07	H3D
Direct Max	42.6%	52.7%	48.0%	61.1%
Direct Sum-sqrt	48.4%	49.0%	51.3%	66.4%
SPP [13]	56.3%	59.5%	67.3%	73.1%
SCFV [16]	61.9%	64.7%	69.0%	<b>76.5%</b>
CL-single	<b>65.8%</b>	<b>72.4%</b>	<b>72.6%</b>	75.3%

**Table 8:** Results obtained by using feature sign quantization.

Dataset	Feature sign quantization	Original
MIT-67	65.2%	65.8%
Birds-200	71.1%	72.4%
Pascal07	71.2%	71.3%
H3D	75.4%	75.3%

other layer of encoding on local features before pooling, the classification accuracy can be greatly boosted. However, in most cases, its performance is still much inferior to the proposed method, as seen in cases of MIT-67, Pascal-07 and Birds-200. The only exception is the result on H3D, where SCFV performs slightly better than our method. However, it needs additional codebook learning and encoding computation while our method does not. Considering this computational benefit and superior performance in most cases, cross-layer pooling is clearly preferable than the other alternative methods.

#### 4.3.3 Feature sign quantization

Finally, we demonstrate the effect of applying a feature sign quantization to pooled feature. Feature sign quantization quantizes a feature to 1 if it is positive, -1 if it is negative and 0 if it equals to 0. In other words, we use 2 bits to represent each dimension of the pooled feature vector. This scheme greatly saves the memory usage. Similar to the above experiment setting, we only report the result on the best performed single resolution setting for each dataset. The results are shown in Table 8. Surprisingly, this coarse quantization scheme does not degrade the performance too much, in three datasets, MIT-67, Pascal-07 and H3D, it achieves almost same performance as the original feature.

## 5. Conclusion

In this paper, we have proposed a new method called cross-convolutional layer pooling to create image representations from the convolutional activations of a pre-trained CNN. Through the extensive experiments, we have shown that this method enjoys good classification performance and

low computational cost. Our discovery suggests that if used appropriately, convolutional layers of a pre-trained CNN contains very useful information and has many advantages over the scheme that using fully-connected layer activations as image representation.

## References

- [1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Proc. Eur. Conf. Comp. Vis.*, 2014.
- [2] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *arXiv:1406.5774*, 2014.
- [3] L. Bourdev, S. Maji, and J. Malik. Describing people: Poselet-based attribute classification. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.
- [4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [5] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan. Hierarchical matching with side information for image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3426–3433, 2012.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.
- [7] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *Proc. Adv. Neural Inf. Process. Syst.*, 2013.
- [8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *Computing Research Repository, arXiv*, abs/1310.1531, 2013.
- [9] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan. Subcategory-aware object classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 827–834, 2013.
- [10] P. W. etc. Caltech-UCSD Birds 200. Technical report, California Institute of Technology, 2010.
- [11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [12] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. *Proc. Eur. Conf. Comp. Vis.*, 2014.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proc. Eur. Conf. Comp. Vis.*, 2014.
- [14] A. Kovashka, D. Parikh, and K. Grauman. Whittlesearch: Image search with relative attribute feedback. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.
- [15] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1106–1114, 2012.
- [16] L. Liu, C. Shen, L. Wang, A. van den Hengel, and C. Wang. Encoding high dimensional local features by sparse coding based fisher vectors. In *Proc. Adv. Neural Inf. Process. Syst.*, 2014.
- [17] T. D. Ning Zhang, Ryan Farrell. Pose pooling kernels for sub-category recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.
- [18] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. *Proc. IEEE Int. Conf. Comp. Vis.*, pages 1307–1314, Washington, DC, USA, 2011. IEEE Computer Society.
- [19] D. Parikh and K. Grauman. Relative attributes. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.
- [20] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 413–420, 2009.
- [21] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *CVPR Workshop*, 2014.
- [22] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.
- [23] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. Eur. Conf. Comp. Vis.*, 2014.
- [24] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *Proc. Eur. Conf. Comp. Vis.*, 2014.

- [25] N. Zhang, R. Farrell, F. Iandola, and T. Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *Proc. IEEE Int. Conf. Comp. Vis.*, December 2013.
- [26] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.
- [27] W. Y. Zou, X. Wang, M. Sun, and Y. Lin. Generic object detection with dense neural patterns and regionlets. In *Proc. British Mach. Vis. Conf.*, 2014.